# Smart Contract Challenge!

November 6, 2023

## 1 Introduction

In this group project for the course, you and your partner will have the opportunity to apply your knowledge and skills by creating a smart contract. The project is open-ended, allowing you to explore various aspects of blockchain technology and develop a smart contract that aligns with your interests.

## 2 Office Hours and Due Date

We are going to hold additional office hours on **Tuesday, 21st(7 - 8)pm** and **Tuesday, 28st(7 - 8)pm**.
The class on **Thursday, 16th** would also serve as office hours from 6 - 8 pm
The deadline for the project is on the last class **Thursday, 30th(6 - 8)pm**
At any point during these sessions you can submit your project for grading.

## 3 Groups

You will be assigned a pair for this project very soon, in order to be paired with another student you will have to fill out this form: Pair Placement form Please ensure you do this by wednesday.

# 4 Project Requirements and Extra Credit Tiers

Your project will be evaluated based on the following requirements:

## 4.1 Requirements (100 points)

1. **Working Smart Contract (40 points)**: Develop a smart contract on Solidity that is well-commented, ensuring clarity and maintainability of the code.

2. **Deployment on Sepolia Testnet (30 points)**: Deploy your smart contract on the Sepolia testnet. Save the transaction hash and provide the URL to Sepolia Blockscanner where the contract was deployed.

3. **Function-by-Function Walkthrough (30 points)**: When presenting to us, provide a detailed walkthrough of your code, explaining each function's purpose and how they interact within the contract. Show us your work on remix or hardhat.

## 4.2 Extra Credit

These are optional features that can serve as a way for you to familiarize yourself with the types of projects our BUIDL team will have in store for you next semester! You can work on however many of these as you like but the extra credit is **capped at 15 points**!

1. **Metamask Transaction Demo (1 points)**: Show us a metamask transaction so that we can get an idea of how users can interact with your smart contract.
2. **OpenZeppelin Standard Implementation (1 points)**: Implement an OpenZeppelin standard like ERC20, ERC721, or another OpenZeppelin standard that wasn't covered in class. Do this by inheriting from the contract and adding your own functionality to it.
3. **User Voting or Staking (1 points)**: Allow users to vote or stake within your smart contract, adding a potential governance factor to your smart contract.
4. **Hardhat Tests with 'Time-Travel' (3 points)**: Write Hardhat tests for your smart contract, particularly using the 'evm_increase_time' feature to test time-dependent functionality.
5. **Hardhat Tests without 'Time-Travel' (1 points)**: If you present and test hardhat unit tests without the increase time feature, you will still be awarded 1 point.
6. **Modifiers Implementation (1 points)**: Replace repeated require statements with well-defined modifiers, enhancing code readability and efficiency.
7. **Contract Inheritance (1 point)**: Implement contract inheritance in your smart contract to demonstrate code re-usability and maintainability by creating a base contract with shared functionality that child contracts can inherit.
8. **Working Frontend (upto 4 points)**: Create a working frontend using JavaScript to interact with your smart contract, providing a user-friendly interface.
9. **Gas-Optimized Code (2 points)**: Optimize your smart contract code for gas efficiency to reduce transaction costs and enhance the overall performance.
10. **Merkle Trees Usage (4 points)**: Implement Merkle trees in your smart contract for efficient verification of large sets of data or proofs, enhancing scalability and security. Make sure to present why you need to use a Merkle Tree.

11. **Oracle Feeds Integration (3 points)**: Integrate oracle feeds into your smart contract to fetch real-world data, enabling decentralized applications to react to external events.

12. **Random Number Generation (2 points)**: Implement a secure and verifiable method for random number generation in your smart contract, which is essential for various applications like gaming and lotteries.

13. **Addressing Real-World Blockchain Challenges (upto 4 points)**: Identify and address a real-world problem using blockchain technology. This may involve solutions related to supply chain management, identity verification, or any other domain where blockchain can provide value.

14. **Top 3 Projects**: The top 3 best projects will get Boiler Blockchain Merch!

# 5 Examples and Resources

To help you get started, here are a few basic examples of smart contracts and resources for developing your project:

- Basic Smart Contract Examples:
  - Voting
  - Auction
  - Staking
  - DAO
  - Chat application
  - Lottery
  - Collectibles
  - Blockchain based game
- Solidity documentation and tutorials
- Patrick Collins Intro to Blockchain.
- Online forums and communities for blockchain developers

# 6 Conclusion

This group project provides you and your team with a practical opportunity to apply your blockchain knowledge and coding skills to create a real-world smart contract. Remember, the key to success is thorough planning, diligent coding, and effective communication of your project's goals and results.

Good luck, and feel free to reach out if you need assistance or have any questions!